AP Computer Science A Notes

mofei w

Hello World

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello World"); // pay close attention to spelling here :)
    }
}
```

Math

Conversions

```
(int) myDouble // double => int via casting!!!
Integer.parseInt(myString) // string => int
Integer.toString(num) // int => string
```

Math Class

```
// static methods; no Math instance is needed

Math.abs(-1) // absolute value => 1
Math.sqrt(16) // square root => 4.0
Math.pow(2, 3) // 2 raised to power of 3 => 8.0
Math.random() // random number => 0 < ? < 1
Math.PI // pi
```

DeMorgan's Law

```
!(a <= b || b > c)

// apply DeMorgan's Law
(a > b && b <= c)
```

Strings

```
int x = 1;
int y = 3;

// string comes first, so the numbers are concatenated
"x + y = " + x + y // x + y = 13

// ints come first, so the numbers are added
x + y + " x + y" // 4 x + y

"" + myInt // hacky way of int => string!!!

String s1 = new String("shaco r");
String s2 = new String("shaco r");

// compares pointers in memory, not the actual values
(s1 == s2) // false

// compares the values of Strings, properly
```

```
s1.equals(s2) // true

// a is starting index, b is ending index PLUS 1
s1.substring(0, 4) // "shac"
```

Arrays

```
// Java arrays can not change lengths after initialization

String[] classes = {"Math", "ELA", "History", "Science", "Art"};

String[] classes = new String[5]; // empty array with 5 slots; default 0/0.0/false/null
classes[0] = "Math";
classes[1] = "ELA";
...

classes.length // length of array
classes[classes.length - 1] // last item of array; adjust by -1 because of 0 start

System.out.println(classes); // this prints out a memory address, not the values

for (int z = 0; z < classes.length; z++) {
    System.out.println(classes[z]);
}

for (String class : classes) {
    System.out.println(class);
}
```

ArrayLists

```
import java.util.ArrayList;

ArrayList<Type> name = new ArrayList<Type>();
name.add("hi"); // add
name.set(0, "hi") // sets the item at position 0
name.remove(1) // removes the item at position 1 - the indexes shift to compensate!!!
```

Control Flow

If

```
if (score == 5) {
    this.isHappy = true;
} else if (score == 4) {
    this.isHappy = true;
} else {
    this.isHappy = false;
}
```

While

```
int z = 0;
while (z < 10) {
    doSomething(); // does this 10 times
    z++;
}
```

For

```
for (int z = 0; z < 10; z++) {
    doSomethingElse(); // also does this 10 times
}
```

Object Oriented Programming

Classes and Inheritance

```
public class StudentAthlete extends Student {
    private String sport;

    public StudentAthlete(String sport)
        // calls constructor of Student class
        super(); // this is the default if no super() call is included

        this.sport = sport;
    }

    // inherits all public instance methods of Student
    // does not inherit constructor

    // inherits instance and class variables
    // usually private so they need to be accessed and modified through methods

    ...
}

public class Student {
    // static variables & functions
    // does not require an instance of the class Student to run!!!
    public static int totalStudents = 0;

    // call with Student.getTotalStudents(); not on an instance
    public static int getTotalStudents() {
        return totalStudents;
    }

    // instance variables
    // private - visable only by this class, NOT the package and the world
    // public - visible to the class, package, and the world
    private String name;
    private double gpa;

    // construcutor
    public Student() {
        this.gpa = 4.0;
        students++;
    }

    // constructor overload - different types, different order, different number of parameters
    // different parameter names will NOT work - method signatures must be different
    // same principle can be applied to methods
    // java will pick the correct one based on the arguments
    public Student(String name) {
        this.name = name;
    }

    // setter methods
    public void setGPA(double gpa) {
        this.gpa = gpa;
    }

    // getter methods
    public double getGPA() {
        return this.gpa;
    }

    // methods
    public void study() {
        // :(
    }

    // abstract method
    // for inheritance - subclasses implement this method
    public abstract void playSport();
}
```

Interfaces

```
public interface Summable {
    public int add(Summable other);

    public int getValue();
}

// implements keyword
public class Book implements Summable {
    ...

    public int getValue() {
        return this.numPages;
    }

    public int add(Summable other) {
        return getValue() + other.getValue();
    }
}
```

Polymorphism

!!!!!!!!!!!!!!!!

References

```
Circle c1 = new Circle("blue");
Circle c2 = new Circle("red");

// c2 & c1 now point to the same object
// the red circle is collected by garbage collector because it is no longer referenced
```

```
c2 = c1;

// sets both c1 & c2 to purple since both point to a single object
c1.setColor("purple");
```

Algorithms

!!!!!!!!!!!!!!